Taylor Series: $\sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n$
With Remainder:
$f(x) = \sum_{n=0}^{m-1} \frac{f^{(n)}(a)}{n!}(x-a)^n + \frac{f^{(m)}(c)}{m!}(x-a)^n$ for some $c \in [x, a]$
$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \dots$
$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots$
$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \dots$
$\ln(1-x) = -x - \frac{x^2}{2} - \frac{x^3}{3} - \frac{x^4}{4} - \dots$
$\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \dots$
$\arctan(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \dots$
Memory
Adds And Multiplies
$P(x) = x^7(a_7 + x^5(a_{12} + x^5(a_{17} + x^5(a_{22} + x^5 \cdot a_27))))$
Binary Exponentiation
Repeating Representation: $\frac{a}{b}$ in base c need
$\frac{a}{b} = \sum d_i c^i + c^j \cdot \frac{d_h}{c^k - 1}$ with $b | c^{h'}(c^k - 1)$ so relevant $\phi(p) = p - 1$ from prime factorisations lead to k and then fractional representation conversion.
IEEE Double Precision Representation For Floating Point Number: 1 Sign Bit 11 Exponent Bits 52 Mantissa Bits, $\epsilon_{\text{machine}} = 2^{-52}$
Chopping, Rounding [Towards 0, Towards Even], Nearest: if the 53rd bit is 0 then round down truncate, if $1.\dots 100\dots 0 \dots = 1.\dots 1\overline{0}$ then round down truncate, else round up add 1 to 52nd bit and reoperate. Can express 9.4 as $+1.0010110011\dots 101 \times 2^3$.

```
Round Down
1.00...000      +
 .       110100 =
1.00...011
Round Down
1.11...111      +
 .       001000 =
1.11...111
Round Up [?] In Register
1.11...111      +
 .       001100 =
1.00...000 x 2^1
```

Relative Error $\frac{|\text{float}(x) - x|}{|x|} \leq \frac{1}{2}\epsilon_{\text{machine}}$ For $x \neq 0$
Bisection Method i.e. Binary Searching: if one has an interval $[a, b]$ which contains a root e.g. $f(a)f(b) < 0$ then iteratedly replace the proper endpoint with $\frac{a+b}{2}$ thereby converging linearly upon the root r with factor $S = \frac{1}{2}$.
Fixed Point: $f(x) = x$
Root Multiplicity: Intuitive $(x-r)^{\text{multiplicity}}$ term in finite polynomial functional representation, degree of derivative where $f^{(\text{multiplicity})}(r) \neq 0$, lowest degree term in Taylor Series around r.
Fixed Point Iteration:
Want Fixed Point Or Root Of $f(x)$ e.g. Fixed Point Of Naive $g(x) = f(x) + x$ Choose Rearrangement Carefully
$x_0 =$ initial guess
$x_{i+1} = g(x_i)$
Linear Convergence
Meaning Errors Satisfy $\lim_{i \to \infty} \frac{e_{i+1}}{e_i} = S = |g'(r)| < 1$
Root Finding Problem
Forward Error: $|r - x_a|$
Backward Error: $|f(x_a)|$
Sensitive, Small Errors In Input Lead To Large Errors In Output, Error Magnification Factor, Condition Number
Sensitivity Formula For Roots: if $\epsilon << f'(r)$, r is a root of $f(x)$ and $r + \Delta r$ is a root of $f(x) + \epsilon g(x)$ then $\Delta r \approx -\frac{\epsilon g(r)}{f'(r)}$.
error magnification factor $= \frac{\text{relative forward error}}{\text{relative backward error}}$
Newton's Method:
$x_0 =$ initial guess
$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$
Quadratically Convergent If Root Has Multiplicity 1 i.e. $f'(r) = 0$.
Meaning Errors Satisfy $\lim_{i \to \infty} \frac{e_{i+1}}{e_i^2} = M, M = \frac{f''(r)}{2f'(r)}$
Otherwise Linear
Multiplicity m Root r:
$\lim_{i \to \infty} \frac{e_{i+1}}{e_i} = S = \frac{m-1}{m}$
Modified
$x_{i+1} = x_i - m \cdot \frac{f(x_i)}{f'(x_i)}$
Quadratically Convergent
Failure
Can blowup diverge to infinity $f(x) = x^{\frac{1}{3}}$ or fail due to divide by 0 if $f'(x_i) = 0$.
Secant Method:
$x_0, x_1 =$ initial guesses
$x_{i+1} = x_i - \frac{f(x_i)(x_i - x_{i-1})}{f(x_i) - f(x_{i-1})}$
Method Of False Position:

```
for i=1,2,3,...
    c=(bf(a)-af(b))/(f(a)-f(b))
    if f(c)=0,stop,end
    if f(a)f(c)<0
        b=c
    else
        a=c
    end
end
```

Muller's Method: use 3 previous points, parabola, nearest root to previous point is next point. Complex arithmetic software complex roots. Faster convergence than Secant Method.
Inverse Quadratic Interpolation: use 3 previous points, parabola function in y, Lagrange Interpolation. Faster convergence than Second Method.
Brent's Method: hybrid method, Matlab fzero command.
Gaussian Elimination: $O(n^3)$
Reduced Row Echelon Form: can augment with b_i to resolve $Ax = b_i$.
Lower Upper Factorisation: add copies of row 1 to rows $2, 3, \dots$ to zero column 1 below the diagonal in U and store those coefficients in column 1 of L, iterate. For the back substitution, solve $Lx' = b_i$ and then $Ux = x'$.
Compute Estimate: $\approx \frac{2}{3} \cdot n^3$ and $2n^2$ operations for Lower Upper Factorisation and each instance of computing x such that $Ax = b_i$ for a total of $\frac{2}{3} \cdot n^2 + 2n^2 k$.
The condition number of a square matrix A, $\text{cond}(A)$, is the maximum possible error magnification factor for solving $Ax = b$, over all right hand sides b.
$\text{cond}(A) = ||A||_\infty \cdot ||A^{-1}||_\infty$.
The matrix norm of an $n \times n$ matrix A is
$||A||_\infty =$ maximum absolute row sum, that is total the

absolute values of each row, and assign the maximum of these n numbers to be the norm of A.
Let x_a be an approximate solution of the linear system $Ax = b$. The residual is the vector $r = b - Ax_a$. The backward error is the norm of the residual $||b - Ax_a||_\infty$, and the forward error is $||x - x_a||_\infty$. The relative backward error is $\frac{||r||_\infty}{||b||_\infty}$ and the relative forward error is $\frac{||x-x_a||_\infty}{||x||_\infty}$. The error magnification error is the ratio of those two, or
error magnification error $= \frac{\text{relative forward error}}{\text{relative backward error}} = \frac{\frac{||x-x_a||_\infty}{||x||_\infty}}{\frac{||r||_\infty}{||b||_\infty}}$.
$PA = LU$ Factorisation [Probably For $n = 2, 3$]: Row Pivoting Swapping To Maximum Magnitude Entry In Column On Diagonal Followed With Usual Tracking Zeroing Of LU
General Reasons For $PA = LU$ Factorisation Over $A = LU$ Factorisation:
Ensures that all multipliers, entries of L, will be no greater than 1 in absolute value. Also solves the problem of 0 pivots. Which are immediately exchanged.
Lagrange Interpolation: $P(x) = \sum y_j \prod_{k \neq j} \frac{x-x_k}{x_j-x_k}$
Theorem 3.3: Assume that $P(x)$ is the (degree $n-1$ or less) interpolating polynomial fitting the n points $(x_1, y_1), \ldots, (x_n, y_n)$. The interpolation error is $f(x) - P(x) = \frac{(x-x_1)(x-x_2)\ldots(x-x_n)}{n!} f^{(n)}(c)$, where c lies in the range i.e. between the smallest and largest of the numbers $x, x_1, \ldots, x_n$.
Chebyshev Interpolation Nodes: On the interval $[a, b]$, $x_i = \frac{b+a}{2} + \frac{b-a}{2} \cos\left(\frac{(2i-1)\pi}{2n}\right)$ for $i = 1, 2, \ldots, n$. The inequality $|(x - x_1)(x - x_2) \ldots (x - x_n)| \leq \frac{\left(\frac{b-a}{2}\right)^n}{2^{n-1}}$ holds on $[a, b]$.
Interpolation Error For Approximating $f(x)$: for nth degree approximation I think it is $|f(x) - Q_n(x)| \leq \frac{|(x-x_1)(x-x_2)\ldots(x-x_{n+1})|}{(n+1)!} \cdot |f^{(n+1)}(c)| \leq \frac{\left(\frac{b-a}{2}\right)^{n+1}}{(n+1)!2^n} \cdot |f^{(n+1)}(c)|$ of course the 6th derivative of $f(x) = e^x$ is simply e^x which is maximised at $x = 1$ for the value of e. And thus one obtains $\frac{1}{6! \cdot 2^5} \cdot e \approx 0.00011798$ so 3 expected correct decimal places after the decimal.
It would seem that for a degree n spline the condition at the joints is that they agree on the 0th, 1st,... ,$n-1$th derivatives. In generality the smoothness vector of desired agreements is defined.
But for cubic splines the properties are also of the form for n given data points $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$:
$S_1(x) = y_1 + b_1(x - x_1) + c_1(x - x_1)^2 + d_1(x - x_1)^3$ on $[x_1, x_2]$
$S_2(x) = y_2 + b_2(x - x_2) + c_2(x - x_2)^2 + d_2(x - x_2)^3$ on $[x_2, x_3]$
$\ldots$
$S_{n-1}(x) =$
$y_{n-1} + b_{n-1}(x - x_{n-1}) + c_{n-1}(x - x_{n-1})^2 + d_{n-1}(x - x_{n-1})^3$ on $[x_{n-1}, x_n]$
2 Point Forward Difference Formula:
$f'(x) = \frac{f(x+h)-f(x)}{h} - \frac{h}{2} f''(c)$ for some $c \in [x, x + h]$.
Generalised Intermediate Value Theorem: let f be a continuous function on the interval $[a, b]$. Let $x_1, x_2, \ldots, x_n$ be points in $[a, b]$ and $a_1, a_2, \ldots, a_n > 0$. Then there exists a number c between a and b such that $(a_1 + a_2 + \cdots + a_n)f(c) = a_1 f(x_1) + a_2 f(x_2) + \cdots + a_n f(x_n)$.
3 Point Centered Difference Formula:
$f'(x) = \frac{f(x+h)-f(x-h)}{2h} - \frac{h^2}{6} f'''(c)$ where $x - h < c < x + h$.
3 Point Centered Difference Formula For 2nd Derivative:
$f''(x) = \frac{f(x-h)-2f(x)+f(x+h)}{h^2} - \frac{h^2}{12} f^{(iv)}(c)$ for some c between $x - h$ and $x + h$.
Extrapolation For Order n Formula: $Q \approx \frac{2^n F(h/2)-F(h)}{2^n-1}$.
This is the extrapolation formula for $F(h)$. Extrapolation, sometimes called Richardson extrapolation, typically gives a higher order approximation of Q than $F(h)$. To understand why, assume that the nth-order formula $F_n(h)$ can be written $Q = F_n(h) + Kh^n + O(h^{n+1})$.
Newton-Cotes Formulas Are Based On Interpolation
a Trapezoid Rule replaces the function with the line interpolating $(x_0, f(x_0))$ and $(x_1, f(x_1))$
b Simpson's Rule uses the parabola interpolating the function at 3 points $(x_0, f(x_0))$, $(x_1, f(x_1))$, and $(x_2, f(x_2))$.
Trapezoid: $\frac{b-a}{2} \cdot (f(a) + f(b))$
Midpoint: $(b - a)f\left(\frac{a+b}{2}\right)$
Simpson: $\frac{b-a}{6}\left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right)$
Composite Trapezoid Rule:
$\int_a^b f(x)dx = \frac{h}{2}\left(y_0 + y_m + 2\sum_{i=1}^{m-1} y_i\right) - \frac{(b-a)h^2}{12} f''(c)$ where $h = \frac{b-a}{m}$ and $c \in [a, b]$.
Composite Midpoint Rule:
$\int_a^b f(x)dx = h\sum_{i=1}^{m} f(w_i) + \frac{(b-a)h^2}{24} f''(c)$ where $h = \frac{b-a}{m}$ and $c \in [a, b]$.
Composite Simpson's Rule: $\int_a^b f(x)dx =$
$\frac{h}{3}\left(y_0 + y_{2m} + 4\sum_{i=1}^{m} y_{2i-1} + 2\sum_{i=1}^{m-1} y_{2i}\right) - \frac{(b-a)h^4}{180} f^{(iv)}(c)$ where $h = \frac{b-a}{2m}$ and $c \in [a, b]$.
Newton Cotes Rule 5.28:
$\int_{x_0}^{x_4} f(x)dx = \frac{4h}{3}(2f(x_1) - f(x_2) + 2f(x_3)) + \frac{14h^5}{45} f^{(iv)}(c)$.
Where $h = \frac{x_4-x_0}{4}$.
Explicit Trapezoid Method:
$w_0 = y_0$
$w_{i+1} = w_i + \frac{h}{2}(f(t_i, w_i) + f(t_i + h, w_i + hf(t_i, w_i)))$
Taylor Method Of Order k:
$w_0 = y_0$
$w_{i+1} = w_i + hf(t_i, w_i) + \frac{h^2}{2} f'(t_i, w_i) + \cdots + \frac{h^k}{k!} f^{(k-1)}(t_i, w_i)$
Midpoint Method
$w_0 = y_0$
$w_{i+1} = w_i + hf\left(t_i + \frac{h}{2}, w_i + \frac{h}{2} f(t_i, w_i)\right)$
Runge-Kutta Method Of Order Four [RK4]:
$w_{i+1} = w_i + \frac{h}{6}(s_1 + 2s_2 + 2s_3 + s_4)$
$s_1 = f(t_i, w_i)$
$s_2 = f\left(t_i + \frac{h}{2}, w_i + \frac{h}{2} s_1\right)$
$s_3 = f\left(t_i + \frac{h}{2}, w_i + \frac{h}{2} s_2\right)$
$s_4 = f(t_i + h, w_i + hs_3)$
Backward Euler Method:
$w_0 = y_0$
$w_{i+1} = w_i + hf(t_{i+1}, w_{i+1})$